

[> home](#) [> about](#) [> feedback](#) [> logout](#)

US Patent & Trademark Office

Citation

The papers of the ACM symposium on Graphic languages [>toc](#)

1976 , Florida, United States

A device independent computer plotting system

Author

Uday G. Gujar

Sponsors

SIGPLAN : ACM Special Interest Group on Programming Languages

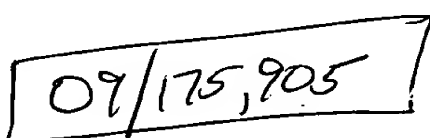
SIGGRAPH : ACM Special Interest Group on Computer Graphics and Interactive Techniques

Pages: 85 - 100 Proceeding-Article

Year of Publication: 1976

[> full text](#) [> abstract](#) [> references](#) [> index terms](#)[> Discuss](#)[> Similar](#)[> Review this Article](#) [Save to Binder](#)[> BibTex Format](#)[↑ FULL TEXT:](#)  [Access Rules](#) [pdf 898 KB](#)[↑ ABSTRACT](#)

This paper describes a computer plotting system which is completely device independent. The user can switch from one plotting device to another without any programming changes. The interface for adding new plotting devices is formalized and discussed. The addition of a new plotting device is completely transparent to the user; in fact, the old programs may be used to produce plots on the new device without any programming changes whatsoever. The system was designed and implemented by the author in late 1971 and has been in use ever since (see Ref. 1). Several new plotting devices, real as well as pseudo, have been added successfully without affecting the users of the system.

[↑ REFERENCES](#)

A DEVICE INDEPENDENT COMPUTER PLOTTING SYSTEM

UDAY G. GUJAR

Computing Centre and
School of Computer Science
University of New Brunswick
Fredericton, N.B., Canada
E3B 5A3

1.0 Introduction:

It is an established fact that the human mind comprehends and understands pictorial information very easily and rapidly. With this experience and the advent of computers, it is little surprise that the computers have been and are being used more and more to produce graphical output.

Almost every computing centre, may it be small, medium or large, has some sort of plotting hardware and the software to support it. More often than not, the software is device dependent. As a result, while switching from one plotting device to another, the user is burdened with several programming changes and forced to live with the associated delays and inconveniences. Further, addition of a new plotting device is a traumatic experience both for the users as well as the system programmers who implement the software for such a device.

This paper describes a computer plotting system which is completely device independent. The user can switch from one plotting device to another without any programming changes. The interface for adding new plotting devices is formalized and discussed. The addition of a new plotting device is completely transparent to the user; in fact, the old programs may be used to produce plots on the new device without any programming changes whatsoever.

The system was designed and implemented by the author in late 1971 and has been in use ever since (see Ref. 1). Several new plotting devices, real as well as pseudo, have been added successfully without affecting the users of the system.

2.0 Devices and Modes:

2.1 Available Plotting Devices:

Plots can currently be obtained on the following plotting devices:

- a. 1403 line printers (set to print m characters per inch and n lines per inch where m and n may have any values).
- b. 611 Tektronix storage oscilloscope.
- c. 1627 Calcomp plotters.
- d. 5100 Gould electrostatic plotter.

It is interesting to note the diverse functional characteristics of the above plotting devices. The 611 storage oscilloscope is an analog device while the 1627 Calcomp plotter is a digital incremental plotter. However, both of these devices allow positive and negative movements in both the X and Y directions. The 5100 electrostatic plotter and line printers have different characteristics compared to the 611 or the 1627 in that they allow paper movement only in one direction. As a result, the programming

technique involved is quite different for these later devices.

The interface between the plotting devices and the main computer (currently IBM 370/158) are also of diversified nature. For example, one 1627 plotter is connected through an IBM 1827 data control unit (which is basically a digital to analog and analog to digital converter) and is situated in the same room as the main computer. However, another 1627 plotter is connected via a HASP remote work station and is situated 130 miles away from the main computer.

Various other plotting devices, for example Hewlett Packard X-Y recorders, have also been connected to the system - though these are not made available to the users because of the non-availability of these devices at the author's installation and such reasons.

2.2 Plotting Modes:

Since the plotting usually requires some manual initial operations involving hardware setup like switching the device on, ensuring plotting pen conditions, readying the interface, etc., two modes of plotting are provided, namely off-line and on-line.

2.2.1 Off-Line Mode of Plotting:

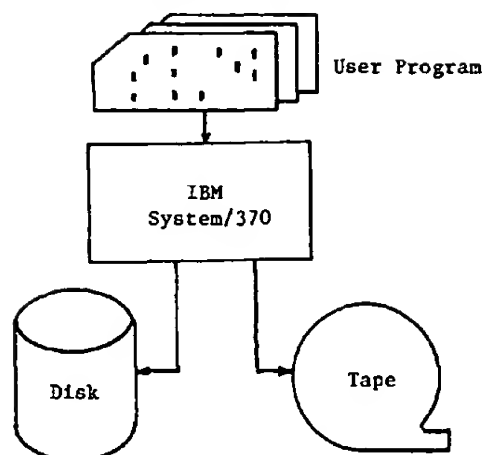
In this mode, the information required to produce a plot is written on a storage medium (e.g. a disk or a tape file) by the plotting package. At some later time, a control program is used to retrieve this information to create the plots on any of the available plotting devices. This process is depicted in Fig. 1.

Off-line mode of plotting can be thought of as a pseudo plotting device. Several types of such pseudo plotting devices are available. These are:

- a. MASSPLOT - This is based on a direct access data set which can hold several plots of several users at the same time. The data set management is handled by the plotting system.
- b. USERPLOT - This enables the user to use his own sequential data set, either on a disk or tape, to store a number of his plots.
- c. HASPPLOT - This uses the HASP spool pack to store the plots and is dependent upon a local modification made to HASP (see Ref. 2).

There is a control program which corresponds to each of these pseudo devices.

Step A: Executed by the user.



Step B: Executed by the Computing Centre

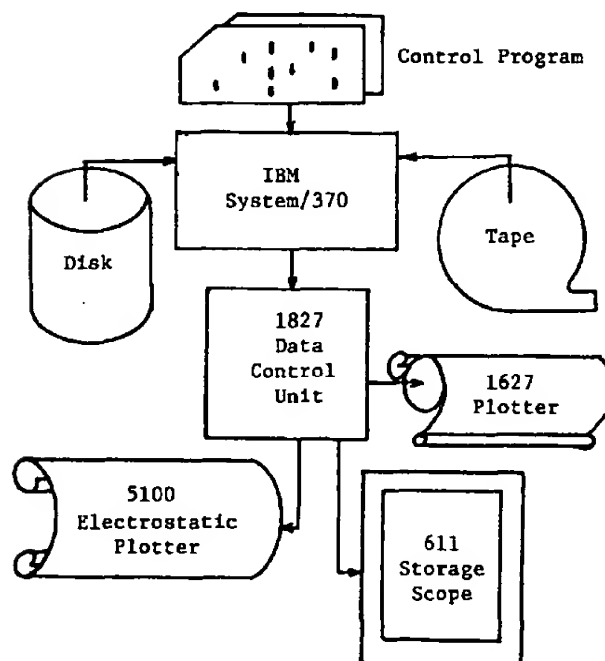


Fig. 1: Off-Line Mode of Plotting

2.2.2 On-Line Mode of Plotting:

In on-line mode, the user program drives the plotting device directly in real time as can be seen from Fig. 2. Since about the only justifiable reason for doing on-line plotting is to permit the programmer to interact directly with the plotting programme, the user must arrange for his job to be run at a special time. He or she would normally supervise the plotting personally.

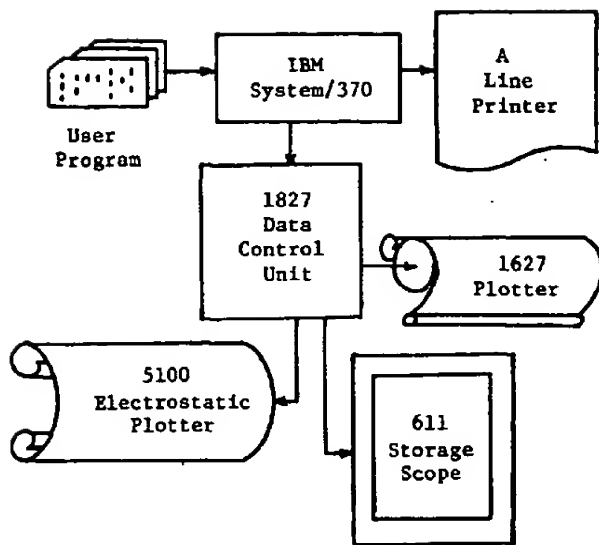


Fig. 2: On-Line Mode of Plotting

2.3 "Simulated" Device:

It is recognized that several installations support the calcomp basic software. To facilitate the transportability of programs from such installations, an interface has been written which simulates basic calcomp software. Using this interface, which is invoked by a single JCL card, the programs that use basic calcomp software can obtain plots on any of the available plotting devices, either directly or through the pseudo plotting devices, without any changes whatsoever.

An interesting corollary of the above is to simulate this plotting system under discussion to generate calcomp basic software calls. Once this is done, the programs written at the University of New Brunswick, using our plotting system, could be sent to any other installation supporting calcomp basic software. It is the author's belief that this can be done rather easily.

3.0 User's Overview:

The system is based on several sub-routines accessible from FORTRAN IV. Effort is made in this section to give an insight into this plotting package. A brief description of all the device independent subroutines is given.

3.1 An Example:

Consider the problem of plotting the equation:

$$y = \sin(x) + \sin(2x) \text{ for } 0 \leq x \leq 2\pi$$

The following program, which is self explanatory, produced the plots given in Figs. 3 and 4.

```

      PI2=2.*3.14159
C-----PLOT A HEADING
C-----{DEFAULT: 1"=1 USER UNIT)
      CALL THICK(0.,5.25,'EXAMPLEn',
*           0.,.5,6,4)
C-----FIX PHYSICAL SIZE TO 4" BY 5"
      CALL AREA(4.,5.)
C-----DEFINE USER UNITS
      CALL SETPLT(0.,-2.,PI2,2.)
C-----DRAW A GRID
      CALL GRID(0.,-2.,PI2,2.,-4.,-4.)
C-----PLACE THE PEN AT INITIAL POINT
      CALL NOWPLT(0,0,0.)
C-----USE * FOR PLOTTING THE CURVE
      CALL PRNTCH('*')
C-----PLOT THE GIVEN EQUATION
      DX=PI2/60.
      DO 1 I=1,60
          X=DX*I
          Y=SIN(X)+SIN(2.*X)
1          CALL NOWPLT(1,X,Y)
C-----IDENTIFY THE CURVE
      CALL CHRPR1(1.4,1.5,
*           '$LY=SIN(X)+SIN(2X)n',0.,.15)
C-----TERMINATE THE PLOT
      CALL ENDPLT
      STOP
      END
  
```

Note that n in the above statements represents a multiple punch of 0, 2 and 8.

3.2 Device Selection:

As far as the user is concerned, he can switch from one plotting device to another by means of a simple change in his job control language (JCL). This often involves changing one JCL card.

All the common plot routines and the routines for generating printer plots are stored in the data set UNB1.FORTLIB. The routines for various pseudo plotting devices are stored in various different libraries, e.g. the routines required for MASSPLOT "device" are stored in UNB1.PLOT.MASSPLOT and those that are required for USERPLOT "device" are stored in UNB1.PLOT.USERPLOT. The routines required for on-line plotting are stored in the data set UNB1.PLOT.REALPLOT.

The proper device is selected by pointing the SYSLIB DD statement of the linkage editor or the loader step to the corresponding data set and UNB1.FORTLIB in that order.

Since most of the users are not very experienced with JCL, several procedures are written for convenience. Thus, for printer plots one may use the following JCL:

```

//...JOB...
// EXEC FORTGCLG
//FORTSYSIN DD *
...
... FORTRAN PROGRAM
...

```

For using MASSPLOT, one simply changes the

```
// EXEC FORTGCLG
```

card to

```
// EXEC MASSPLOT
```

Similarly for using USERPLOT, above card is changed to

```
// EXEC USERPLOT
```

and a data definition card (DD card) is provided to give the attributes of the user's data set which might be:

```
//GO*FT23F001 DD DISP=SHR,DSN=MYFILE
```

3.3 Optional Arguments:

Several routines in this package have a variable number of arguments. Many of these are completely written in FORTRAN (see Appendix B) using the technique developed in Ref. 3. Very briefly, from the user's point of view, the user may specify as many of these optional arguments as desired. The arguments which are omitted get the predefined default values. Arguments can only be omitted from the right hand end of the list. For example, consider the following routine:

```
CALL ELLIPS(XC,YC,RMAJOR[,RMINOR,THETA,FROMTH,TOTH])
```

where last four arguments are optional. The following calls are valid:

```

CALL ELLIPS(XC,YC,RMAJOR)
CALL ELLIPS(XC,YC,RMAJOR,RMINOR)
CALL ELLIPS(XC,YC,RMAJOR,RMINOR,THETA)
CALL ELLIPS(XC,YC,RMAJOR,RMINOR,THETA,FROMTH)
CALL ELLIPS(XC,YC,RMAJOR,RMINOR,THETA,FROMTH,TOTH)

```

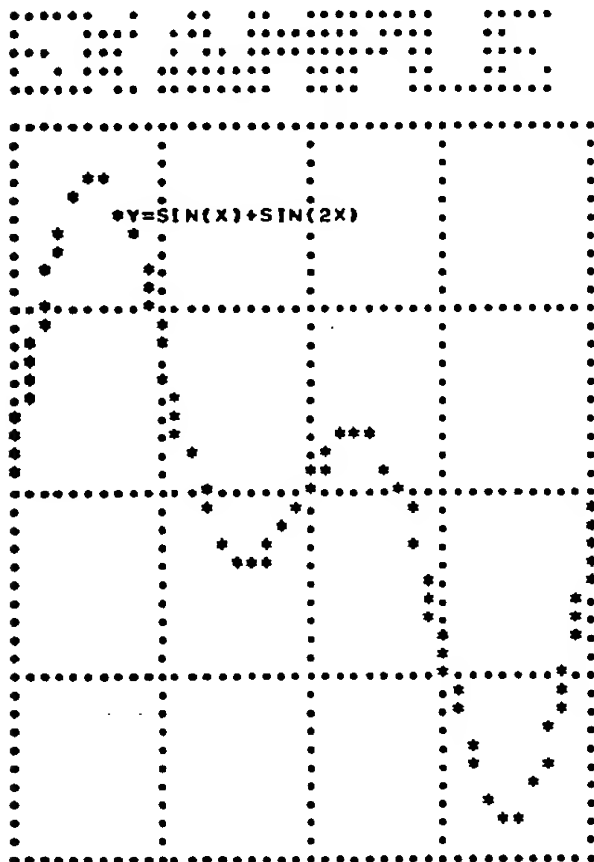


Fig. 3: Plot Produced on a Line Printer

EXAMPLE

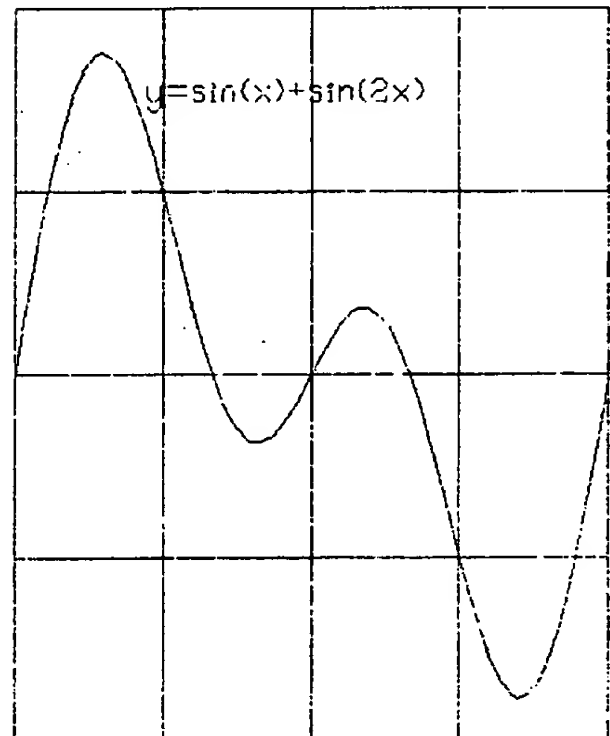


Fig. 4: Plot Produced on the 5100 Electrostatic Plotter

Note that if a special value for THETA is required, a value for RMINOR must be specified even if the default value of RMINOR is suitable. Similarly, to be able to specify TOTH, the values for RMINOR, THETA and FROMTH must be supplied in that order.

3.4 Common Plot Routines and Their Functions:

All the common plot routines have been categorized as far as possible and a brief description of their usage is given in this section. See Fig. 5 for a pictorial overview. The user may use any or all of these routines for plotting on any device. If any particular routine is not applicable, by virtue of its peculiarity, to some plotting device a dummy routine is automatically supplied. This eliminates the necessity of making changes, which otherwise might have been required, to the deck while switching from one plotting device to another. See Appendix A for the summary of calling sequences of all the routines and Ref. 1 for the complete details. Appendix B contains such details as the memory required for each routine, the language each routine is written in, entry points and the other routines called.

3.4.1 Initialization Routines:

These routines should be called, if at all, before any other plot routine. However, one may call the subroutines AREA and SETPLT several times to achieve some fascinating results. The following is a list of the initialization routines:

- DEVICE:** Enables the user to choose between the plotting devices (e.g. 611, 1627 or 5100) during the on-line mode of plotting.
- AREA:** Defines the physical size of the plot.
- SETPLT:** Enables the user to define his own units for plotting rather than forcing some arbitrary units.
- PLOTID:** Enables the user to supply an 80 character identification label to his plot when using the off-line mode of plotting. This identification label, which may contain some special instructions, is conveyed to the plotter operator by the plotting system just before the initiation of the start of the actual plot.

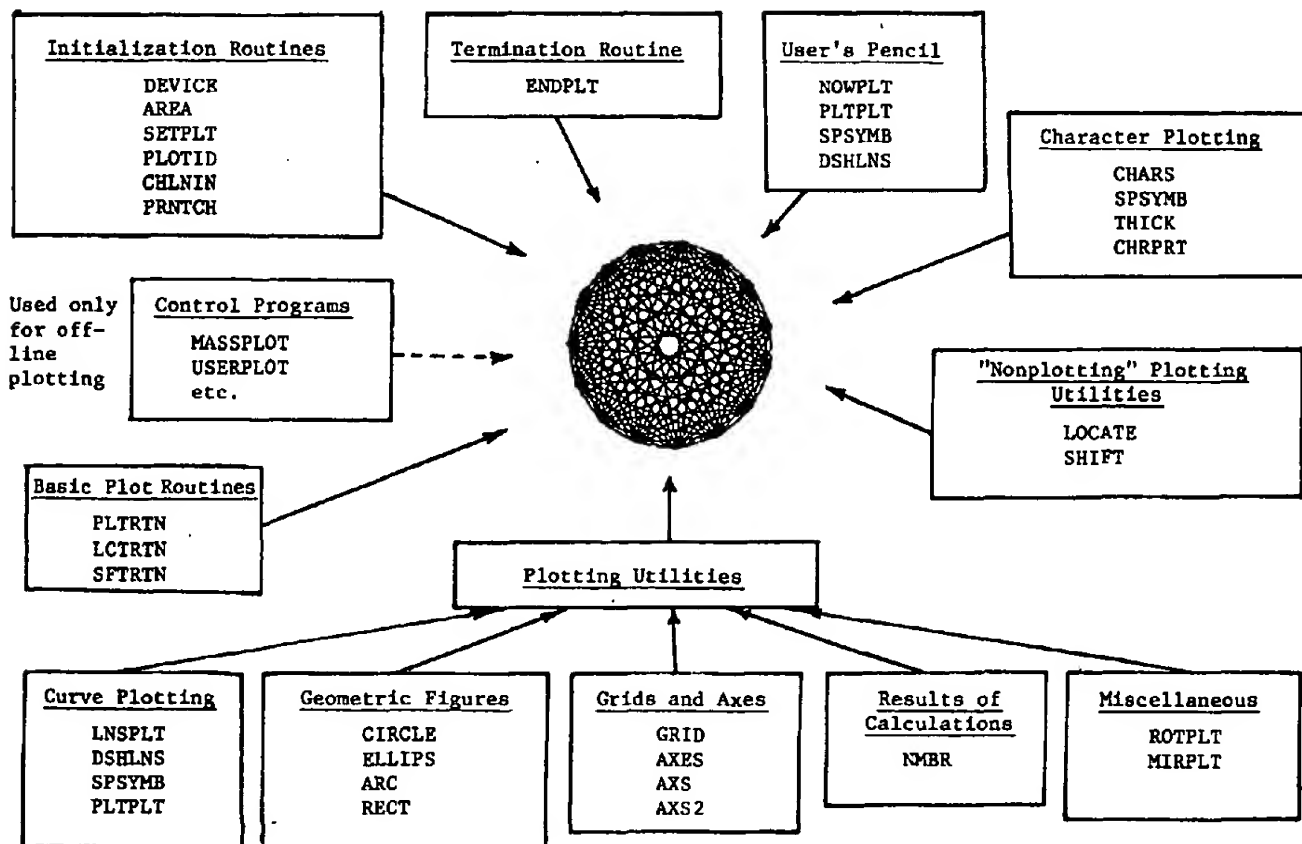


Fig. 5: Overview of the Plotting Package

- e. CHLNIN: Defines the number of characters per inch and the number of lines per inch for the line printer.
- f. PRNTCH: Enables the user to select a print character for plotting on a line printer; this character may be changed as often as required as the plot progresses.

3.4.2 Termination Routine:

There is one termination routine, namely ENDPLT. All the plotting jobs must call this subroutine ENDPLT at least once to terminate the current plot frame and thus force completion of the plot on the desired plotting device.

3.4.3 The User's Pencil:

The following routines are provided to move the plotting pencil:

- a. NOWPLT: Moves the plotting pencil from its current position to a specified location in a straight line either in up or down position. This permits the user to draw a point or a straight line or to position the plotting pencil.
- b. PLTPLT: Enables the user to make several sets of plotting pencil movements, as specified above in "NOWPLT", in one call.
- c. SPSYMB: See Section 3.4.4.
- d. DSHLNS: See Section 3.4.6.

3.4.4 Character Plotting:

A variable width character generator has been written which enables the user to plot a string of characters containing all the EBCDIC graphic symbols which include upper and lower case letters, several special symbols, digits etc. In addition, several useful plotting symbols are also available. Special control characters are provided for writing both upper and lower case letters, superscripts, subscripts and for starting new lines. The following is a list of these routines:

- a. CHARS: Plots a string of characters with any specified height at any specified angle.
- b. SPSYMB: Plots one of fifteen special symbols with specified height at any angle. The user may optionally join all or any of these points to obtain one or more curves.
- c. THICK: Enables the user to plot characters with specified thickness, height and inclination.

- d. CHRPR: PRINTS, rather than plots, a string of characters on a line printer. For other devices, this routine simply calls 'CHARS' to PLOT the characters.

3.4.5 "Nonplotting" Plotting Utilities:

There are two routines which do not produce plots but are useful:

- a. LOCATE: Returns the current position of the plotting pencil.
- b. SHIFT: Enables the user to shift, by a specified amount, all future plotting with respect to plotting done prior to the call to SHIFT.

3.4.6 Plotting Utilities:

It is anticipated that some of the sections of plots required by users will be recurring in nature. Some examples are to draw axes, grids, circles, ellipses, to obtain curves joining several points, etc. Several routines have been written to fulfill this anticipated need, to avoid possible duplication of efforts and to provide the user with an easy and convenient tool.

A. Curve Plotting:

- a. LNSPLT: Draws straight lines between several specified points in one call.
- b. DSHLNS: Joins a set of points with dashed lines. The number and lengths of dashes and gaps are under program control, thus enabling the user to draw any type of dashed lines.
- c. SPSYMB: See Section 3.4.4.
- d. PLTPLT: See Section 3.4.3.

B. Geometric Figures:

- a. CIRCLE: Enables the user to plot a circle or a circular arc with a specified centre and radius.
- b. ELLIPS: Plots an ellipse or an elliptical arc with specified radii, centre and inclination.
- c. ARC: Draws a circular arc, or a circle, given the co-ordinates of the centre and the beginning and "end" of the arc.
- d. RECT: Plots a rectangle of specified size with specified inclination.

C. Grids and Axes:

- a. GRID: Plots a linear, semi-log or log-log grid of any size, anywhere on the plotting area with any required density.
- b. AXES: Plots X-Y axes perpendicular to each other anywhere on the plotting area, crossing at a specified point, with linear, semi-log or log-log tick marks which could be of any specified lengths.
- c. AXS: Plots a line, of specified length, at any angle with equally spaced tick marks of any specified lengths and properly centred annotation of any specified size on a specified side of the line.
- d. AXS2: Plots a line, between two specified points, with equally spaced tick marks of any specified lengths and properly centred annotation of any specified size on a specified side of the line.

D. Results of Calculations:

- a. NMBR: Plots REAL, INTEGER, LOGICAL or COMPLEX number(s) of any length in a specified FORTRAN FORMAT at a specified location with any specified height and inclination.

E. Miscellaneous:

- a. ROTPLT: Facilitates plotting with rotated X-Y axes.
- b. MIRPLT: Draws the mirror image of a complex plot by placing an imaginary mirror at any desired angle.

3.4.7 Basic Plot Routines:

The X-Y co-ordinates of all the points given to the routines discussed so far are in the user's units defined by the parameters of the subroutines AREA and SETPLT. These parameters are referenced between the subroutines through a labelled COMMON P611G. All these plotting routines directly or indirectly invoke the plot routines which handle X-Y points in one thousandths of an inch. The formulae which define the relations between X,Y in user units and IX,IY in one thousandths of an inch, in terms of various parameters in the labelled COMMON P611G, are:

$$IX = (X - XMIN) / XSCALE * XINCH$$

$$IY = (Y - YMIN) / YSCALE * YINCH$$

$$X = IX / XINCH * XSCALE + XMIN$$

$$Y = IY / YINCH * YSCALE + YMIN$$

where,

$\left. \begin{matrix} XMIN \\ YMIN \end{matrix} \right\}$ - Minimum X and Y values in user units.

$\left. \begin{matrix} XSCALE \\ YSCALE \end{matrix} \right\}$ - Lengths of the plotting area in X and Y directions in the user units.

$\left. \begin{matrix} XINCH \\ YINCH \end{matrix} \right\}$ - Physical dimensions, is one thousandths of an inch, in X and Y directions.

The general user seldom has a need to call these basic routines which are:

- a. PLTRTN: Moves the plotting pencil in up or down position with X-Y co-ordinates specified in one thousandths of an inch.
- b. LCTRTN: Returns the co-ordinates, in one thousandths of an inch, of the current position of the plotting pencil.
- c. SFTRTN: Enables the user to shift, by a specified amount in one thousandths of an inch, all future plotting with respect to plotting done prior to the call to SFTRTN.

4.0 Backbone of the System:

All the routines discussed so far are device independent in the sense that the user may call them while plotting on any of the plotting devices - pseudo or real. Now an attempt will be made to describe how this device independence is achieved.

There are four plotting functions which depend upon the nature of the plotting device, namely,

- a. physical plotting,
- b. location of the current position of the plotting pencil,
- c. termination of a plot, and
- d. shifting plots.

The above functions are achieved by the subroutines PLTRTN, LCTRTN, ENDPLT and SFTRTN respectively. (To be precise, LCTRTN, ENDPLT and SFTRTN are entry points of PLTRTN.) Though function b listed above is not strictly device dependent, it is convenient and straightforward to associate it with physical plotting.

All the plotting routines which produce plots call the subroutine PLTRTN directly or indirectly. Thus, the device independence of the entire plotting package, as far as a choice of the plotting device is concerned, is based on the availability of different versions of the subroutine PLTRTN (with entry points LCTR TN, ENDPLT and SFTRTN).

4.1 Selection of a Device:

A typical FORTRAN job in IBM System 370 goes through the following distinct processes (see Ref. 4 and 5):

- a. Compilation,
- b. Linkage editor (or loader) and
- c. Execution.

During the first process, the FORTRAN program (including the supplied subprograms) is translated into machine language to produce what is called an "object module". The calls made to the non-supplied subprograms are generated as external references and are marked as unresolved.

During the linkage editor (or loader) process, all the external references are resolved in the following manner:

- a. All the object modules given as input to the linkage editor (or loader) are merged with the object module produced as a result of the compilation.
- b. All the remaining external references are resolved from the supplied libraries (specified through the SYSLIB data definition card). These libraries are searched in the given order. It is important to note that the search for a particular routine terminates as soon as it is found.

The module produced as a result of the second process is referred to as a "load module". During the last process, the load module is executed.

Several versions of the subroutine PLTRTN (with entry points LCTR TN, ENDPLT and SFTRTN) have been written and stored in various different libraries. The proper version of PLTRTN is obtained by providing suitable concatenation of the data sets for library search through the SYSLIB DD card for the linkage editor or the loader. This process can be conveniently automated for a common user by creating appropriate JCL procedures and storing them in the system procedure library (SYS1.PROCLIB).

It follows from the above discussion that the addition of a new plotting device,

whether it be a real or pseudo, is also a simple matter of writing the subroutine PLTRTN (with entry points LCTR TN, ENDPLT and SFTRTN) for that particular device. There need not be any change whatsoever either to the user's programs or to the rest of the plotting package.

Testing of a new plotting device is a very straightforward process since the system programmer can supply the PLTRTN as a source program - either FORTRAN or assembler, as an object module or as the load module. This testing can proceed concurrently with the other user programs using other plotting devices without any interaction.

4.2 Various Plotting Devices:

It would be beyond the length of this paper to describe all the plotting devices that have been and are connected to the system. Therefore, only a few major ones will be discussed in this section.

In the off-line mode of plotting, the X-Y points are stored as ICNT, IX and IY where ICNT is the control byte and IX, IY are the coordinates of a point in one thousandths of an inch. The meanings of the control byte are:

ICNT	ACTION
0	move plotting pencil to (IX,IY) in up position
1	lift plotting pencil up, then move it to (IX,IY) and put it down; i.e. plot the point (IX,IY)
2	draw a straight line from previous plotting pencil position to (IX,IY) and leave the plotting pencil down
3	do as ICNT=2 above and lift the plotting pencil up
4	call ENDPLT; ignore (IX,IY)
5	call SFTRTN with (IX,IY) as the arguments.

The plotting package arranges ICNT such that $0 \leq \text{ICNT} \leq 5$.

4.2.1 Off-Line Plotting Through MASSPLOT:

PLTRTN for this version is stored in the library UNBL.PLOT.MASSPLOT. This version stores the plots produced by the users in a predefined and formatted direct access data set. The user is never required to create the data set. The plotting package stores several plot frames for several users in this data set which consists of 255 blocks each containing 748 bytes. The first block is the directory block and the remaining are plot-data blocks.

An entry is created in the directory for each plot frame stored. The format of

this directory entry is given in Fig. 6. The first byte contains the plot frame number starting from 11 and the last byte contains the block number where the plot-data for this plot frame begins. Plot code consists of 11 packed decimal digits giving the time of the day the plot frame was created. The first byte of the plot code is made hexadecimal FF (=255₁₀) when the plot is deleted.

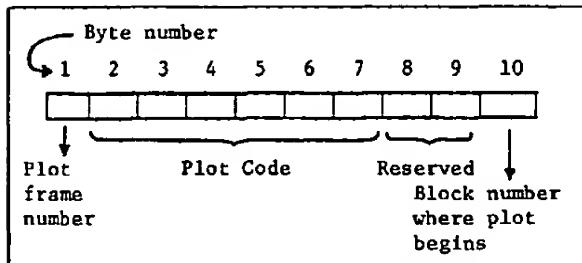


Fig. 6 Directory Entry for MASSPLOT

The format of the data structure used for the plot-data blocks is given in Fig. 7.

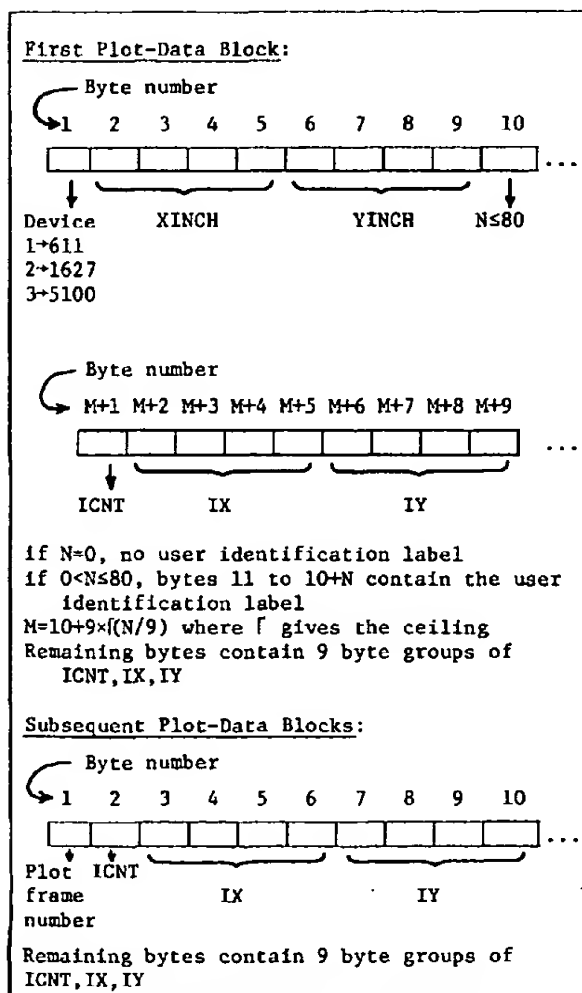


Fig. 7 Plot-Data Blocks for MASSPLOT

The control program gives the plotter operator various options such as producing the plot frame on the specified device, skipping the plot frame, deleting the plot frame, forcing the plot frame on any of the available real devices, clearing the entire data set, etc.

4.2.2 Off-Line Plotting Through USERPLOT:

PLTRTN for this version is stored in the data set UNBL·PLOT·USERPLOT. It stores plots for a particular user in a sequential data set created by him. Since the data set can contain the plot frames for only one user, no directory is required.

Each record is 126 bytes long and has the format shown in Fig. 8. The only restriction on the data set is that the record length must be 126 bytes. It may be blocked and may be created on any type of available auxiliary storage device (e.g. a disk or a tape).

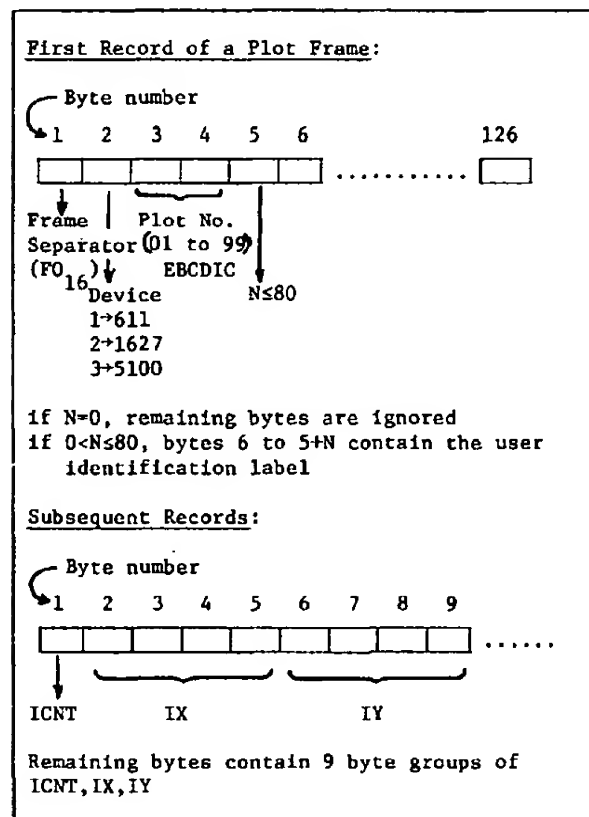


Figure 8 Data Structure for USERPLOT

The same data structure is used for HASPPLOT - in fact, USERPLOT has been created by a very simple and straightforward simulation of HASP local modifications. That is precisely the reason why the record length is so small.

As in the MASSPLOT version, the control program used to retrieve the plots stored in USERPLOT system gives the plotter operator various options such as producing the plot frame on the specified device, skipping the plot frame, forcing the plot frame on any of the available real plotting devices, etc.

4.2.3 On-Line Mode (REALPLOT):

PLTRTN for this version is stored in the library UNB1.PLOT.REALPLOT. This is the version that actually produces plots.

As created at the author's installation, this version gives access to the following three devices:

- a. 611 storage oscilloscope,
- b. 1627 X-Y plotter and
- c. 5100 electrostatic plotter

The choice is governed through a full word integer in the labelled COMMON P611G.

Basically, the IX,IY values passed to PLTRTN or SFTRTN are scaled to the proper units for the selected device and these are then passed on to the driver routines (see Section 5.0) which do the plotting. ENDPLT simply invokes the appropriate driver routine. Since the information required for LCTRNT is already in PLTRTN itself, no corresponding driver routines are required.

The control programs for MASSPLOT and USERPLOT invoke this version of PLTRTN to produce the actual plots for the user.

It is perhaps quite obvious by now that one could create versions of PLTRTN which plot only on the 611 or the 1627 or the 5100 and store them in three different libraries. Then, the device selection can be achieved even in the on-line mode of plotting by a simple JCL change.

4.2.4 Printer Plots:

PLTRTN for this version is stored in the library UNB1.FORTLIB. Since this is the default library searched for resolving external reference for the linkage editor or loader process in all the FORTRAN procedures at the author's installation, the printer plots can be produced without any special JCL cards.

The version currently used in production builds the plot in a two dimensioned array which is printed when ENDPLT is called. This forces the arbitrary limits on the size of the plot that can be produced on the printer.

Several other versions have been written and successfully used which remove the arbitrary restriction on size and which use various different algorithms.

5.0 Device Dependent Routines:

Though the device dependence may be handled in PLTRTN itself, it is often convenient, and indeed sometimes desirable in order to access several real plotting devices during the execution of a program, to localize the device dependency in the driver routines. These driver routines may then be called by PLTRTN.

5.1 Drivers:

The driver routines are:

Pxxxx - movement of plotting pencil

Sxxxx - shifting of plots

Exxxx - termination of plots

where, xxxx is the device. PLTRTN calls Pxxxx, SFTRTN calls Sxxxx and ENDPLT calls Exxxx.

Thus, the routines for the 611 storage oscilloscope are:

```
CALL P611(IC,IX,IY)
CALL S611(IX,IY)
CALL E611
```

where IX and IY are in 611 absolute units, i.e. between 0 and 32742.

The routines that drive 1627 and 5100 are:

```
CALL P1627(IC,IX,IY)
CALL S1627(IX,IY)
CALL E1627
CALL P5100(IC,IX,IY)
CALL S5100(IX,IY)
CALL E5100
```

where IX and IY are in one hundredths of an inch.

5.1.1 Tektronix 611 Storage Oscilloscope:

This is a storage cathode ray tube with a physical picture area of 6.375 inches in the X direction and 8.125 inches in the Y direction. The full scale deflection is produced by applying one volt to the X and/or Y direction. Since the 611 is controlled by the IBM 370/158 computer through the 1827 data control unit, the number 32742 generates one volt and creates the full scale deflection.

The algorithm used to generate straight lines is an extension of symmetrical digital differential analyzer (ESDDA).

Assume that the beam can store its path if up to MAXINC increment is given in the X and/or Y direction. MAXINC may be typically 64 or even higher. Then the beam can be moved in MAXINC different ways in any octant. See Fig. 9 which depicts various possible movements for different values of MAXINC. By choosing the movement that is closest to the desired straight line, one can obtain a fairly good approximation.

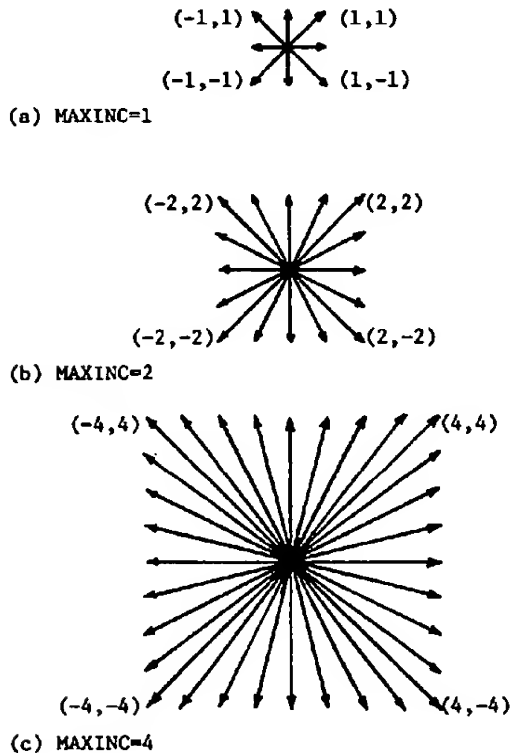


Fig. 9: Possible Movements for Various Values of MAXINC

The flowchart of the algorithm, based on this idea to draw a straight line from (X_0, Y_0) to (X, Y) is given in Fig. 10. The following notation is used in Fig. 10:

$|d| \leftrightarrow$ absolute value of d
 $\text{REM}(d, x) \leftrightarrow$ Remainder of $d \div x$
 $\text{SIGN}(d) \leftrightarrow$ sign of d

The reader might recognize it as the generalized symmetrical digital differential analyzer, the later being a special case when MAXINC is one.

5.1.2 Calcomp 1627 X-Y Plotter:

As is perhaps well known, the Calcomp 1627 plotter works in steps of

one-hundredths of an inch and as such the drivers work in that unit.

The algorithm used is the one given in Ref. 6. Concise mathematical definition of this algorithm appears in Appendix C.

5.1.3 Gould 5100 Electrostatic Plotter:

The algorithm developed for this device is the subject matter of another technical report (see Ref. 7).

5.2 Odds and Ends:

There are certain other types of device dependencies introduced by the peculiar nature of the plotting device. For example, subroutines PRNTCH, CHLNIN and CHRPT are required for line printers. It would be a great inconvenience, and indeed a great nuisance, if the user were required to insert and remove certain statements from his program depending upon the plotting device. Appropriate dummy subroutines, requiring sometimes as few as two bytes, are written and placed in proper libraries to take care of such situations.

6.0 Plot Accounting:

The method used for charging the plots produced is given in this section. This method, which was designed for the 611 storage scope and the 1627 X-Y plotter, may undergo some modifications for the 5100 electrostatic plotter.

The effective plot time (EPT) is calculated according to the following formula:

$$\text{EPT} = A * \text{PCLS} + B * \text{PINCH} \text{ milliseconds}$$

where,

PCLS - Number of calls to PLTRTN

PINCH - Plot inches

A - milliseconds required to lift the plotting pencil up or put it down

B - milliseconds required to move the plotting pencil one inch

PCLS is a good approximation of up and down movements of the plotting pencil.

PINCH is calculated according to the following formula:

$$\text{PINCH} = \sum (|X - X_0| + |Y - Y_0|)$$

where (X_0, Y_0) and (X, Y) are the starting and end points of each plotting pencil movement and the summation is over all motions used to produce a plot frame.

The values of constants A and B depend upon the plotting device. The current values are:

Device	A	B
611	.1	17
1627	140	500

The plotting charge (PC) is given by the following formula:

$$PC = C \cdot EPT + D \text{ units}$$

where C and D are constants which are currently set at 0.00025 and 40 respectively.

7.0 Conclusions:

A powerful computer plotting system has been presented. The system has been in use for a number of years and has proved to be satisfactory to several users.

Several new plotting devices have been added since the inception of the system. These additions have been completely transparent to the users.

The aspects that are dependent on the plotting device and those that are independent of the plotting device have been separated and identified. Based on this, the interface for adding a new plotting device has been formalized and defined. The implementation of this interface has been straightforward and has worked very satisfactorily from the system programmer's point of view.

The package is open ended in the sense that it can be expanded both in the device independent section as well as device dependent section. For example, addition of a contouring package or three dimensional plotting once implemented will work for all the plotting devices.

As an example of a new pseudo plotting device, one may consider the implementation of two dimensional transforms (see Ref. 8) to obtain the effects of translation, rotation, scaling, etc. Further, one could incorporate the windowing, clipping, etc.

7.0 Acknowledgements:

The author wishes to thank Mr. B.J. Claus and Dr. D.M. Fellows for interesting and productive discussions. The addition of the 1627 and the 5100 plotter became possible because of the hardware interface, provided by Mr. B. Kurz and Dr. W.D. Wasson, between the 1827 data control unit and these devices. Software for interfacing the 5100 into the plotting system was written by Mr. J.A. Fitzgerald. The co-operation of Mr. L. Barton and his crew in

running hundreds of programs on the computer is appreciated. Thanks are due to Mr. D.M. Miller for reading the manuscript of this paper and to Mrs. A. Stoczek for typing it.

References:

1. Gujar, U.G., 'Computer Plotting', Computing Centre, University of New Brunswick, Fredericton, N.B., Canada, 212 pages, 1972; second printing Dec. 1974; third printing Oct. 1975.
2. Gujar, U.G., 'The RJE/RJO interface with HASP 4.0', (to be published).
3. GUJAR, U.G., 'Subroutines with variable number of arguments', TR75-004, School of Computer Science, University of New Brunswick, Fredericton, N.B., Canada, April 1975. Also presented at the Fifth Annual Conference on Numerical Mathematics and Computing, University of Manitoba, Winnipeg, Canada, Oct. 1975.
4. 'FORTRAN IV (G and H) Programmer's Guide', IBM, order no. GC28-6817-3.
5. 'OS/VS Linkage Editor and Loader', IBM, order no. GC26-3813-1.
6. Bresenham, J.E., 'Algorithm for computer control of a digital plotter', IBM Systems Journal, Vol. 4, No. 1, 1965.
7. Fitzgerald, J.A. and Gujar, U.G., 'Programming electrostatic plotter type devices', (to be published).
8. Newman, W.M. and Sproull, R.F., 'Principles of Interactive Computer Graphics', McGraw-Hill, 1973.

Appendix A: Summary of Calling Sequences of the Routines

In the following list, the optional arguments are enclosed in square brackets. The routines marked with an asterisk (*) are the drivers and as such are device dependent; all other routines are device independent.

```

CALL ARC(XC,YC,XS,YS[,XL,YL])
CALL AREA(XLNGTH,YLNGTH)
CALL AXES(XMN,YMN,XX,YY,XCROSS,YCROSS,
  XD,YD[,TICKLN])
CALL AXS(X,Y,IARRAY,SIZE,THETA,VALUE,
  VALINC,NDD[,HIGHTL,HIGHTN,TICKLN])
CALL AXS2(X1,Y1,X2,Y2,IARRAY,VALUE,
  VALINC,NDD[,HIGHTL,HIGHTN,TICKLN])
CALL CHARS(X,Y,IARRAY[,THETA,HEIGHT,
  XRET,YRET])
CALL CHLNIN(XCHARS,YLNS)
CALL CHRPRT(X,Y,IARRAY[,THETA,HEIGHT,
  XRET,YRET])
CALL CIRCLE(XC,YC,RADIUS[,FROMTH,TOTH])
CALL DEVICE(ID)
CALL DSHLNS(XX,YY[,DSHGPL,IDG,ILIM,
  ISTART,IINCR])
CALL ELLIPS(XC,YC,RMAJOR[,RMINOR,THETA,
  FROMTH,TOTH])
CALL ENDPLT
*CALL E1627
*CALL E5100
*CALL E611
CALL GRID(XMN,YMN,XX,YY,XD,YD)
CALL LCTRNT(IX,IY)
CALL LNSPLT(XX,YY[,ILIM,ISTART,IINCR])
CALL LOCATE(X,Y)
CALL MIRPLT(IC,X,Y,THETA[,XMIR,YMIR])
CALL NMBR(X,Y,NUMBER,FORMAT[,THETA,HEIGHT,
  ILIM,ISTART,IINCR,XRET,YRET])
CALL NOWPLT(IC,X,Y)
CALL PLOTID(IARRAY)
CALL PL7PLT(ICC,XX,YY[,ILIM,ISTART,IINCR])
CALL PLTRTN(IC,IX,IY)
CALL PRNTCH(ICH)
*CALL P1627(IC,IX,IY)
*CALL P5100(IC,IX,IY)
*CALL P611(IC,IX,IY)
CALL RECT(X,Y,XL[,YL,THETA])
CALL ROTPLT(IC,XROT,YROT,THETA[,X,Y])
CALL SETPLT(XMN,YMN,XX,YY)
CALL SFTRTN(IX,IY)
CALL SHIFT(X,Y)
CALL SPSYMB(X,Y,I[,THETA,HEIGHT,XRET,YRET])
*CALL S1627(IX,IY)
*CALL S5100(IX,IY)
*CALL S611(IX,IY)
CALL THICK(X,Y,IARRAY,THETA,HEIGHT,IDX,IDY)

```

Appendix B: Libraries and Plotting Routines

- Notes: 1. P611G referred to in the following tables is a labelled COMMON block.
 2. F and A in the column "written in" of the following tables stand for FORTRAN IV and 370 Assembler respectively.

A. Library: UNB1-FORTLIB

Routines	Entry Points	Memory (bytes)	Other Routines Called	Written In
ARC	-	1058	VARARG PLTRTN ATAN2 SQRT COS SIN P611G	F
AREA	-	334	P611G	F
AXS	AXS2	3892	VARARG CORE IBCOM# CHRPRT PLTRTN ATAN2 SQRT COS SIN P611G	F
CHARS	SPSYMB	2864	PLTRTN COS SIN P611G	A
CIRCLE	-	946	VARARG PLTRTN COS SIN P611G	F
DEVICE	-	2	-	A
DSHLNS	-	1698	VARARG LOCATE PLTRTN SQRT P611G	F
ELLIPS	-	1188	VARARG PLTRTN COS SIN P611G	F
GRID	AXES	2720	FIXPR# FIXPI# NOWPLT LOCATE VARARG ALOG10 P611G	F
LOCATE	-	424	LCTRNT P611G	F
MIRPLT	-	732	VARARG PLTRTN COS SIN P611G	F

A. Library: UNB1·FORTLIB (continued)

Routines	Entry Points	Memory (bytes)	Other Routines Called	Written In
NMBR	-	1188	VARARG CORE IBCOM# CHRPRT P611G	F
NOWPLT	-	458	PLTRTN P611G	F
PLOTID	-	2	-	A
PLTPLT	LNSPLT	444	PLTRTN P611G	A
PLTRTN	CHRPRT CHLNIN ENDPLT LCRTTN PRNTCH SFTRTN	4082	IBCOM# VARARG COS SIN P1403G P611G	F
RECT	-	1266	VARARG PLTRTN COS SIN P611G	F
ROTPLT	-	708	VARARG PLTRTN COS SIN P611G	F
SETPLT	-	394	P611G	F
SHIFT	-	428	SFTRTN P611G	F
THICK	-	1898	CHARS COS SIN P611G	F

B. Library: UNB1·PLOT·MASSPLOT

Routine	Entry Points	Memory (bytes)	Other Routines Called	Written In
CHRPRT	-	12	CHARS	A
DEVICE	-	124	P611G	A
PLTRTN	ENDPLT LCRTTN PLOTID SFTRTN	2840	P611G Acctng. modules	A
PRNTCH	CHLNIN	2	-	A

C. Library: UNB1·PLOT·USERPLOT

Routine	Entry Points	Memory (bytes)	Other Routines Called	Written In
CHRPRT	-	12	CHARS	A
DEVICE	-	124	P611G	A
PLTRTN	ENDPLT LCRTTN PLOTID SFTRTN	2532	WTL IBCOM# P611G Acctng. modules	F
PRNTCH	CHLNIN	2	-	A

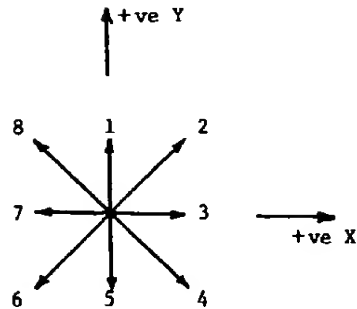
D. Library: UNB1·PLOT·REALPLOT

Routine	Entry Points	Memory (bytes)	Other Routines Called	Written In
CHRPRT	-	12	CHARS	A
DEVICE	-	124	P611G	A
PLOTID	-	2	-	A
PLTRTN	ENDPLT LCRTTN SFTRTN	1518	P611 P1627 P5100 E611 E1627 E5100 S611 S1627 S5100 Acctng. modules	F
PRNTCH	CHLNIN	2	-	A
P1627	E1627 S1627	1052	-	A
P5100	E5100 S5100	7728	SORT	A
P611	E611 S611	1852	-	A

Appendix C: Algorithm Used for the 1627

The following is a concise mathematical definition of the algorithm by Bresenham (Ref. 6) to draw a straight line from (X0,Y0) to (X,Y) on the 1627 X-Y plotter.

If the possible plotter movements are:



then the algorithm is:

1. Compute DX and DY as

$$DX = X - X_0 \text{ and } DY = Y - Y_0$$

2. Determine DA and DB as

$$DA = \text{Maximum of } |DX| \text{ and } |DY|$$

$$DB = \text{Minimum of } |DX| \text{ and } |DY|$$

3. Calculate DEL as $2 \cdot DB - DA$

4. Compute M1 and M2 as

$$\begin{aligned} M1 &= 7 \text{ if } |DX| \geq |DY| \text{ and } DX < 0 \\ &= 3 \text{ if } |DX| \geq |DY| \text{ and } DX \geq 0 \\ &= 1 \text{ if } |DX| < |DY| \text{ and } DX \geq 0 \\ &= 5 \text{ if } |DX| < |DY| \text{ and } DX < 0 \end{aligned}$$

$$\begin{aligned} M2 &= 2 \text{ if } |DX| \geq 0 \text{ and } DY \geq 0 \\ &= 4 \text{ if } |DX| \geq 0 \text{ and } DY < 0 \\ &= 6 \text{ if } |DX| < 0 \text{ and } DY < 0 \\ &= 8 \text{ if } |DX| < 0 \text{ and } DY \geq 0 \end{aligned}$$

5. If $DEL \geq 0$, execute the plotter movement M2 and modify DEL as:

$$DEL = DEL + 2 \cdot DB - 2 \cdot DA$$

If $DEL < 0$, execute the plotter movement M1 and modify DEL as:

$$DEL = DEL + 2 \cdot DB$$

6. Repeat Step 5 until you reach (X,Y).